REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

AD-A264 601

1. AGE	3. REPORT TYPE AND DATES COVERED Reprint
4. TITL Title shown on Reprint	5. FUNDING NUMBERS DAAL03-91-C-0043
6. AUTHOR(S) Authors listed on Reprint	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stanford U Stanford, CA 94305	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ARO 29210.2-EL
11. SUPPLEMENTARY NOTES The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

ABSTRACT SHOWN ON REPRINT

93 5 05 016

93-09724

3PX

14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

B. Herndon, A. Raefsky, R.J.G. Goossens, and R. W. Dutton,
"PISCES-MP - Adaptation of a Dusty Deck for Multiprocessing,"
Proceedings of NASECODE-VII, May 19-22, 1992.

PISCES MP - Adaptation of a Dusty Deck for Multiprocessing

Bruce P. Herndon, Arthur Raefsky, and Ronald J.G. Goossens
Integrated Circuits Laboratory, Stanford University, Stanford, CA 94305

I. INTRODUCTION

Scalable multiprocessors offer high performance with relatively low cost. Unfortunately, the programming model required to take advantage of these architectures is a radical departure from traditional paradigms. Most users are unwilling to discard the knowledge and expertise captured by existing dusty-deck programs in exchange for a faster yet unproved and unfamiliar parallel code. To explore the potential for providing vastly improved dusty-deck performance while preserving the knowledge implicit in the program, we have parallelized the device simulator PISCES [1] on an Intel iPSC/860™ hypercube.

Section II gives a brief overview of PISCES. Section III describes the methods used to transform PISCES into a parallel code. A demonstration of the computational power of the new parallel device solver is presented in Section IV. Improvements to the linear solver are discussed in Section V. Finally, conclusions are given in Section VI.

II. Overview of PISCES

PISCES is a two-dimensional device simulator consisting of approximately 40,000 lines of FORTRAN-77. Code development has been ongoing throughout the last ten years, involving several generations of graduate students and researchers. Although the program structure is rather inelegant, great care has been taken to validate the code as well as to improve and calibrate the physical models. It solves Poisson's equation and the continuity equations below:

$$\nabla(\epsilon\nabla\Psi) = -q(p - n + N_D^+ - N_A^-) - \rho_F$$

$$\frac{\partial n}{\partial t} = \frac{1}{q}\nabla \bullet J_n - U_n$$

$$\frac{\partial p}{\partial t} = \frac{1}{q}\nabla \bullet J_p - U_p$$

The equations are discretized using irregular triangular grid and are solved using either Newton or Gummel nonlinear schemes. A large number of physical models are supported. The sparse systems of linear equations arising from these methods are solved using an optimized sparse direct solver as described in [2]. Figure 1 shows the major code components. Lucas observed in [3] that for even small simulation grids, PISCES spends between 77% and 96% of its runtime solving the coupled nonlinear device equations. The nonlinear solver

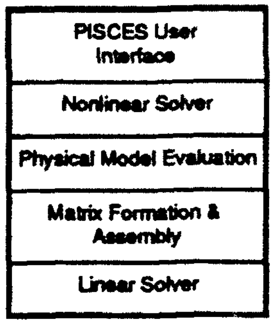

Figure 1
Major elements in PISCES

repeatedly forms element matrices, assembles a global matrix, solves the resulting sparse system, and updates the nonlinear solution. Recent experience shows nonlinear solution times grow to be more than 99% of the runtime for moderate to large grid sizes. The remaining fraction of time is spent in the user interface (UI) parsing user input, performing I/O, and generating grid.

III. Parallelization of PISCES

Typical PISCES simulations require several hours on moderate grid sizes and days on large grids. Clearly, significant performance gains would be welcomed by users. Restructuring the nonlinear solver and all of its requisite routines to run in parallel would breathe new life into the simulator. However, the UI is inherently serial and must be treated differently. In order to accommodate this dichotomy, we split the code into two programs. Figure 2 shows the structure of PISCES MP. The bulk of PISCES MP runs on the hypercube including all code for nonlinear solution, model evaluation, matrix formation, matrix assembly, and linear solution. Although we left the majority of PISCES code untouched, many changes were necessary. Fortunately, changes rarely pervaded the entire code. For instance, we were forced to add data structures to map each processor's local domain into the global simulation grid and to determine each processor's responsibility for shared portions in each domain. We were also forced to modify those physical models and assembly routines that relied on non-local information. For example, all grid points attached to an electrode must be given a consistent potential value. If these grid points are distributed across multiple processors, the processors must communicate to determine the proper value. Finally, we replaced the linear direct method, each processor directly eliminates all local equations and updates a dense block corresponding to the shared equations. Rather than solve the dense shared block directly, we use the preconditioned generalized minimal residual (GMRES) algorithm [5]. GMRES requires less global data traffic than a direct method. Table 2 compares the linear solution times on the 9200 grid point example described earlier. The solution times for a single linear solution are given. This simulation required in excess of 120 linear solutions. Not surprisingly, the fully direct method is faster for a small number of processors due to the large amount of local computation coupled with the small amount of necessary data transfer. As expected, for larger numbers of processors the hybrid method outperforms the fully direct method by reducing the amount of shared data transfer. This allows for the exploitation of greater concurrency and results in faster overall sclution times.

	Hybrid	Direct
Computational Unit	Time (s)	Time (s)
iPSC/860 4 CPU	11.023	9.604
iPSC/860 8 CPU	6.821	5.618
iPSC/860 16 CPU	3.942	4.819
iPSC/860 32 CPU	3.768	6.951

Table 2
Comparison of linear solution times on 9200 grid point example

VI. Conclusions

In this paper, we have described the parallelization of PISCES. We have retained the valuable expertise captured in the long-term development of the program. Our initial results show significant performance gains. In fact, the program not only runs existing simulations faster but also provides the capability of solving vastly larger problems than originally feasible. We have also addressed the communication bottleneck created by the direct solver when using large numbers of processors. We have implemented a hybrid solver that produces greater parallel efficiency in these cases.

Acknowledgments

We would like to thank Horst Simon for providing the spectral nested dissection code and Bob Lucas for improving our understanding and implementation of the multi-frontal method. This research was sponsored by the State of California's Department of Commerce under grant number C90-072 and by DARPA.

References

[1] M.R. Pinto, C.S Rafferty, H.R. Yeager, and R.W. Dutton, "PISCES-IIB Supplementary Report," Stanford Electronics Lab., Stanford Univ., Stanford, CA, 1985.

[2] D.A. Calahan and P.G. Buning, "Vectorized General Sparsity Algorithms with Backing Store," SEL Report 96, Systems Engineering Laboratory, University of Michigan, Ann Arbor, MI, 1977

[3] R.F. Lucas, "Solving Planar Systems of Equations on Distributed-Memory Multiprocessors," Ph.D. Dissertation, Stanford Univ., Dec., 1987.

[4] A. Pothen, H.D. Simon, and K.P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," SIAM J. Mat. Anal. Appl., 11 (1990), pp. 430-452

[5] Y. Saad and M.H. Shultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," SIAM J. Sci. Stat. Comp., vol. 7, pp. 856-869, 1986.